Improved DIRECT-type Algorithm Based on a New Approach for Identification of Potentially Optimal Hyper-Rectangles

Naziheddine Belkacem , Lakhdar Chiter * , Mohammed Louaked

Posted Date: 29 November 2023

doi: 10.20944/preprints202311.1873.v1

Keywords: Global optimization; Derivative-free global optimization; Diagonal partitioning scheme; DIRECT-type algorithms; Potentially optimal hyper-rectangles

Article

Improved DIRECT-type Algorithm Based on a New Approach for Identification of Potentially Optimal Hyper-Rectangles

Naziheddine Belkacem [1], Lakhdar Chiter [2,*] and Mohammed Louaked [3]

[1] Affiliation 1; Department of Mathematics, Faculty of Mathematics and Informatics, University of Bordj Bou Arreridj El-Anasser, 34030, Algeria
[2] Affiliation 2; Department of Mathematics, Ferhat-Abbas University, Fundamental and Numerical Mathematics Lab. (LMFN); Ferhat-Abbas University of Sétif1, Sétif 19000, Algeria
[3] Laboratoire de Mathématiques Nicolas Oresme, Université de Caen, 14000 Caen, France
* Correspondence: lchiter@univ-setif.dz

Abstract: This paper introduces new improvements to the modified version of the BIRECT (BIsecting RECTangles) algorithm referred to as BIRECTv. We explore various approaches, by first including a grouping strategy for hyper-rectangles having almost the same sizes by categorizing them into different classes. Therefore constraining them not to exceed a certain pre-defined threshold (a small positive value to define the tolerance level). This approach allows for more efficient computation and can be particularly useful when dealing with a large number of hyper-rectangles with varying sizes. To avoid over-sampling, and preventing redundant or excessive sampling, at some shared vertices in descendant subregions, a particular vertex database is used to limit the number of samples taken within each subregion to two. The experimental investigation shows that these improvements have a positive impact on the performance of the BIRECTv(imp.) algorithm and the proposal is a promising global optimization algorithm compared to the original BIRECTv algorithm and its variants. Additionally, the BIRECTv(imp.) algorithm showed particular efficacy in solving high-dimensional problems.

Keywords: global optimization; derivative-free global optimization; diagonal partitioning scheme; DIRECT-type algorithms; potentially optimal hyper-rectangles

MSC: 90C56; 90C26

1. Introduction

In scientific and engineering domains, optimization problems frequently involve objective functions that can only be obtained through "black-box" methods or simulations, and they often lack explicit derivative information. In black-box optimization cases, the development of derivative-free global optimization methods (DFGO) has been forced by the need to optimize various and often increasingly complex problems in practice because the analytic information about the objective function is unavailable [10,15,33–37]. The absence of derivative information requires the use of derivative-free global optimization (DFGO) methods. DFGO techniques are specifically designed to optimize functions when derivatives are unavailable or unreliable. These methods explore the function's behavior by sampling it at various points in the input space.

This paper considers a global optimization problem

$$\min_{\mathbf{x}\in D} f(\mathbf{x}), \tag{1}$$

that require only the availability of objective function values but no derivative information, therefore numerical methods using gradient information can not be used to solve problem (1). The objective function is supposed to be Lipschitz-continuous with some fixed but unknown Lipschitz constant,

 doi:10.20944/preprints202311.1873.v1

and the feasible domain is an n-dimensional hyper-rectangle $D = [\mathbf{l}, \mathbf{u}] = \{\mathbf{x} \in \mathbb{R}^n : l_j \leq x_j \leq u_j, j = 1, \ldots, n\}$.

Global optimization approaches can be categorized into two main types: deterministic [1,5,6,26,28] and stochastic methods [12,42]. These methods address optimization problem (1) using various domain partition schemes, often involving hyper-rectangles [5,25,42]. While many DIRECT-type techniques employ hyper-rectangular partitions, other alternative approaches use simplicial partitioning [18,19] (as DISIMPL-C [16] and DISIMPL--V [17]) or diagonal sampling schemes (see [22,23,25], as adaptive diagonal curves [24]).

DIRECT-type algorithms, such as the DIRECT (DIvide RECTangles) algorithm [7–9] are the most widely used partitioning-based algorithms for global optimization problems. One of the challenges faced by these algorithms is the selection of potentially optimal rectangles (the most promising), which can lead to inefficiencies and increased computational costs. In this paper, we provide a comprehensive review of techniques and strategies aimed at reducing the set of selected potential optimal hyper-rectangles in DIRECT-type algorithms. We explore various approaches, including a novel grouping strategy which simplify the identification of hyper-rectangles in the selection procedure. This strategy consists in rounding or approximating the measurements (sizes) of hyper-rectangles, that are extremely small in size, by grouping them together into classes. This simplification can help in various computational or analytical tasks, making the problem more manageable without significantly compromising the accuracy of the analysis or optimization process.

Our review highlights the importance of reducing the number of function evaluations while maintaining the algorithm's convergence properties. The recent papers by [29,38,39] provides a good and a comprehensive overview of techniques aimed at reducing the set of potentially optimal rectangles in DIRECT-type algorithms. It significantly contributes to the field of derivative-free global optimization and serves as a valuable resource for researchers and practitioners seeking to enhance the efficiency and effectiveness of such algorithms. Some suggested methods are summarized in [9,21,29,36,38].

In the context of Optimization Methods in Engineering Mathematics, the size of a hyper-rectangle often incorporate constraints imposed by the engineering problem. These constraints ensure that the optimization process adheres to real-world limitations, such as physical boundaries, safety margins, or resource constraints. For example, in structural engineering, the size of a hyper-rectangle could represent the permissible ranges for material properties, dimensions, or loads. In engineering optimization, reducing the size of a hyper-rectangle can represent the imposition of stricter constraints. This ensures that the optimized solution adheres to more stringent requirements, such as safety limits or design specifications.

We also use an additional assumption to improve this version allowing to evaluate the objective function only once at each vertex of each hyper-rectangle. The objective function values at vertices could be stored in a special vertex database, and then the result is directly retrieved from this database when required. In addition, an update to the modified optimization domain is applied for some test problems as used in the previous version [3].

The original DIRECT algorithm faces challenges when it comes to sampling points at the edges of the feasible region, which can slow down its convergence, particularly in cases where the best solution is located at the boundary. This limitation is especially pronounced in constrained problems. Recent research [13,32,38] has emphasized the importance of addressing this issue, showing that it's possible to achieve faster convergence by employing strategies that sample points at the vertices of hyper-rectangles, especially when solutions are near the boundary.

Taking these insights into account, we've integrated one of the latest versions of DIRECT-type algorithms into our approach, a new diagonal partitioning and sampling scheme called BIRECTv (BIsection of RECTangles with Vertices) based on the BIRECT algorithm. In the BIRECTv framework, the objective function is evaluated at specific points within the initial hyper-rectangle. Instead of evaluating the objective function only at the vertices, as done in most DIRECT-type algorithms, BIRECTv samples two points along the main diagonal of the initial hyper-rectangle, located at $1/3$ and 1 of the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

way along the diagonal. This approach provides more comprehensive information about the objective function, and helps to improve convergence, particularly near boundaries.

The contributions of the paper can be summarized as follows:

1. A review of techniques and strategies aiming to reduce the set of selected potential optimally hyper-rectangles in DIRECT-type algorithms.
2. Introduction of a novel grouping strategy which simplify the identification of hyper-rectangles in the selection procedure in DIRECT-type algorithms.
3. The new approach incorporates a particular vertex database to avoid more than two samples in descendant subregions.
4. The improvements of BIRECTv algorithm positively impacted the performance of the BIRECTv algorithm.

The rest of this paper is organized as follows. In Sect. 2.1, a review of the original BIRECT algorithm is provided, while in Sect. 2.2 a brief description of the new sampling and partitioning scheme called BIRECTv algorithm is also discussed. In Sect. 2.3, we incorporate a novel scheme for grouping and selecting potential optimal hyper-rectangles in BIRECT-type algorithms. Numerical investigation and discussion of the results is given in Sect. 3. Finally, in Sect. 4, we conclude the paper and outline potential directions for future prospects.

2. Materials and Methods

This section provides an overview of the original BIRECT algorithm and its modifications.

2.1. The original BIRECT

The BIRECT (BIsection of RECTangles) algorithm, developed in [20], employs a diagonal space-partitioning approach and involves two primary procedures: sampling on diagonals and bisection of hyper-rectangles.

In the initialization step, the algorithm begins by evaluating the objective function at two initial points, "lower" $\mathbf{l} = (l_1, \ldots, l_n) = (1/3, \ldots, 1/3)^T$ and "upper" $\mathbf{u} = (u_1, \ldots, u_n) = (2/3, \ldots, 2/3)^T$, positioned along the main diagonal of the normalized domain, considered as the first unit hyper-cube, $\bar{D} = \bar{D}_0^1 = [\bar{\mathbf{l}}, \bar{\mathbf{u}}] = \{\bar{\mathbf{x}} \in \mathbb{R}^n : 0 \leq \bar{l}_j \leq \bar{\mathbf{x}} \leq \bar{u}_j \leq 1, j = 1, \ldots, n\}$. The hyper-cube representing the search space is then divided into a set of smaller hyper-rectangles obeying a specific sampling and partitioning scheme using the following critera (see Algorithm 1).

2.1.1. Selection criteria

- At each iteration (kth iteration), starting from the current partition

$$\mathcal{P}_k = \{\bar{D}_k^i : i \in \mathbb{I}_k\},$$

  where $\mathbb{I}_k$ is the index set identifying the current partition, a new partition $\mathcal{P}_{k+1}$ is created by bisecting a set of potentially optimal hyper-rectangles from the previous partition.
- The identification of a potentially optimal hyper-rectangle is based on lower bound estimates of the objective function over each hyper-rectangle, with a fixed rate of change $\tilde{L} > 0$ (analogous to a Lipschitz constant).
- A hyper-rectangle $\bar{D}_k^j$, $j \in \mathbb{I}_k$ is considered potentially optimal if specific inequalities involving ε (a positive constant) and the current best-known function value $f_{\min}$ are satisfied.

$$\min\left\{f(\mathbf{l}^j), f(\mathbf{u}^j)\right\} - \tilde{L}\delta_j^k \leq \min\left\{f(\mathbf{l}^i), f(\mathbf{u}^i)\right\} - \tilde{L}\delta_i^k, \quad \forall i \in \mathbb{I}_k \tag{2}$$

$$\min\left\{f(\mathbf{l}^j), f(\mathbf{u}^j)\right\} - \tilde{L}\delta_j^k \leq f_{min} - \varepsilon|f_{min}|, \tag{3}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

where the measure (distance, size) of the hyper-rectangle is given by

$$\bar{\delta}_k^i = \frac{2}{3}\|\bar{\mathbf{b}}^i - \bar{\mathbf{a}}^i\|, \tag{4}$$

A hyper-rectangle $\bar{D}_k^j$ is potentially optimal if the lower bound for f computed by the left-hand side of (2) is optimal for some fixed rate of change $\tilde{L}$ among the hyper-rectangles of the current partition $\mathcal{P}_k$. Inequality (3) helps prevent excessive emphasis on local search [7].

2.1.2. Division and sampling criteria

- After the initial partitioning, BIRECT proceeds to future iterations by partitioning potentially optimal hyper-rectangles and evaluating the objective function $f(x)$ at new sampling points.
- New sampling points are generated by adding and subtracting a distance equal to half the side length of the branching coordinate from the previous points. This approach allows for the reuse of old sampled points in descendant subregions.
- An important aspect of the algorithm is how the selected hyper-rectangles are divided. For each potentially optimal hyper-rectangle, the set of maximum coordinates (edges) is computed, and the hyper-rectangle is bisected along the coordinate (branching variable x_{br}, $1 \leq br \leq n$) with the largest side length. The selection of the coordinate direction is based on the lowest index j, prioritizing directions with more promising function values.

$$br = \min\left\{\underset{1\leq j\leq n}{\arg\max} = \left\{d_j^i = \left|b_j^i - a_j^i\right|\right\}\right\}, \tag{5}$$

The partitioning process continues until a predefined number of function evaluations has been performed, or a stopping criterion is satisfied. The algorithm keeps track of the best (smallest) objective function value $f(\bar{\mathbf{x}})$ found over all sampled points in the final partition. The corresponding generated point $\bar{\mathbf{x}}$ at which this value was achieved provides an approximate solution to the optimization problem. The main steps of the BIRECT algorithm are outlined in Algorithm 1 (see [20] for a detailed pseudo-code).

The BIRECT algorithm is a robust optimization technique that efficiently explores the search space, combines global and local search strategies, and strives to find the optimal or near-optimal solution for multidimensional optimization problems. For a more comprehensive understanding, additional details can be found in the original paper [20].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

Algorithm 1 Main steps of BIRECT algorithm

Input : Objective function: f, search-space: D, tolerance: ϵ_{pe}, the maximal number of function evaluations: M_{max}, and the maximal number of iterations: K_{max};
Output : The best objective function value: f_{min}, global minimizer: x_{min}, and algorithmic performance measures: m, k and pe (if needed);
Normalize the search space D to the unit hyper-cube $\bar{D}$;
Initialize $\mathbf{l}^1 = (1/3, \ldots, 1/3)$ and $\mathbf{u}^1 = (2/3, \ldots, 2/3)$, evaluate $f(\mathbf{l}^1)$ and $f(\mathbf{u}^1)$, and set $f_{min} = min\left\{f(\mathbf{l}^1), f(\mathbf{u}^1)\right\}$, $x_{min} = \underset{x \in \{\mathbf{l}^i, \mathbf{u}^i\}}{\operatorname{argmin}} f(x)$, $m = 2$, $k = 1$, $\mathbb{I}_k = \{1\}$;
while $pe > \varepsilon_{\text{pe}}$ and $m < \text{M}_{\max}$ and $k < \text{K}_{\max}$ **do**
    Identify the index set $\mathbb{P}_k \subseteq \mathbb{I}_k$ of potentially optimal hyper-rectangles (POHs);
    Set $\mathbb{I}_k = \mathbb{I}_k \backslash \{\mathbb{P}_k\}$;
    for $i \in \mathbb{P}_k$ **do**
        Select the branching variable **br** (coordinate index) Eq. (5);
        Divide $\bar{D}^i$ into a two new hyper-rectangles $\bar{D}^{m+1}$ and $\bar{D}^{m+2}$. Update δ_{m+1} and δ_{m+2};
        Create the new sampling points $\mathbf{l}^{m+1}$ and $\mathbf{u}^{m+2}$;
        Update the partition set: $\mathcal{P}_k = \mathcal{P}_k \backslash \bar{D}_k^j \cup \bar{D}_k^{m+1} \cup \bar{D}_k^{m+2}$;
        if $f_{min}^{m+1} \leq f_{min}$ **or** $f_{min}^{m+1} \leq f_{min}$ **then**
            Update f_{min} and x_{min};
        end if
        Update performance measures: k, m and pe;
    end for
end while
Return : f_{min}, x_{min} and algorithmic performance measures: m, k and pe.

2.2. Description of the BIRECTv Algorithm

In this subsection, we return back to one of the most recent versions of DIRECT-type algorithms (called BIRECTv) developed in [3]. One effective strategy is to sample points at the vertices of the hyper-rectangles. This approach ensures that points near the boundaries are explored, increasing the chances of finding solutions located there. Sampling at vertices can significantly improve convergence when the optimal solution is at or near the boundary, see [32]. A description of two different partitioning schemes used in DIRECT-type algorithms is shown in Figure 1. The original DIRECT algorithm primarily focuses on sampling within the interior of the feasible region, which means it may miss exploring points near the boundary. Therefore, it may require a large number of iterations to converge to the optimal solution. This slow convergence is because it relies on subdividing hyper-rectangles within the interior, and it may take many iterations before a hyper-rectangle boundary coincides with the solution. The studies conducted in [33,40] have indeed highlighted the significant impact of the limitation in convergence when the optimal solution lies at the boundary of the feasible region. This issue is particularly prevalent in constrained optimization problems, where solutions often lie at the boundary due to the constraints imposed on the variables.

However, a challenge arises when the newly created sampling points coincide with previously evaluated points at shared vertices. This leads to additional evaluations of the objective function, increasing the number of function evaluations per iteration. To address this issue, the paper suggested modifying the original optimization domain to obtain a good approximation of the global solution.

This approach was presented as an alternative to locate solutions that are situated near the boundary. The results of the experiments demonstrated that the proposed modification to the optimization domain positively impacted the performance of the BIRECTv algorithm. It outperformed the original BIRECT algorithm and the two popular DIRECT-type algorithms on the test problems. Additionally, the BIRECTv algorithm showed particular efficacy in solving high-dimensional problems.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

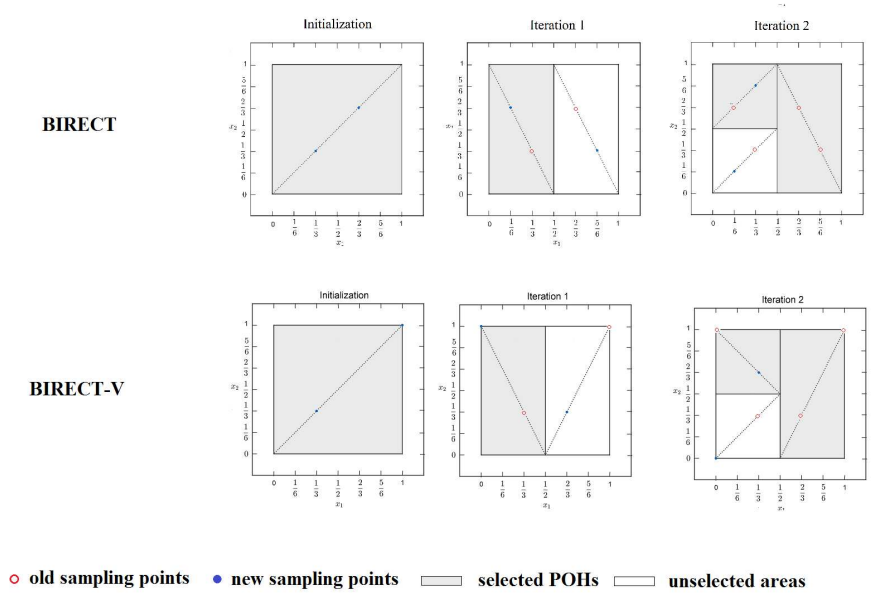

Figure 1. Description of the initialization and the first two iterations used in two different sampling and partitioning schemes (BIRECT: upper figure), and (BIRECTv: lower figure) on a two-dimensional example.

2.3. Integrating Scheme for Identification of Potentially Optimal Hyper-rectangles in DIRECT-based Framework

In this section, we introduce an innovative grouping technique that streamlines the hyper-rectangle identification process during selection. This approach involves the rounding or approximation of measurements (sizes) for hyper-rectangles of exceedingly small dimensions. These are then organized into classes, yielding simplification that enhances the manageability of computational and analytical tasks. Importantly, this simplification doesn't substantially impact the precision of the analysis or optimization process. The selection of the most promising hyper-rectangles in DIRECT-type algorithms is a crucial aspect of optimization.

Various strategies have been developed to enhance this selection process, resulting in different versions of the algorithm. In the DIRECT-l variant [2,9], the size of a hyper-rectangle is measured by the length of its longest side, which corresponds to the infinity-norm. This approach allows DIRECT-l to group more hyper-rectangles with the same measure, resulting in fewer distinct measures. Moreover, in DIRECT-l, only one hyper-rectangle from each group is selected, even if there are multiple potentially optimal hyper-rectangles in the same group. This reduces the number of divisions within a group. DIRECT-l is found to perform well for lower-dimensional problems that do not have an excessive number of local and global minima.

The aggressive version of DIRECT takes a different approach by selecting and dividing a hyper-rectangle of every measure in each iteration. While this strategy requires more function evaluations compared to other versions of DIRECT, it may be advantageous for solving more complex problems. The PLOR algorithm simplifies the set of potentially optimal hyper-rectangles to just two: the maximal and the minimal Lipschitz constants. This reduction allows the PLOR approach to be independent of user-defined parameters. It strikes a balance between local and global search during the optimization process by considering only these two extreme cases.

In two-phase globally and locally biased algorithms, the selection procedure during one of the phases operates similarly to the original DIRECT algorithm, considering all hyper-rectangles from the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

current partition. However, in the second phase, the selection of potentially optimal hyper-rectangles is constrained based on their measures. Globally-biased versions [17,24] focus on larger subregions, addressing the algorithm's first weakness, while locally-biased versions [2,14] concentrate on smaller subregions, addressing the second weakness of DIRECT-type algorithms. These adaptations and strategies aim to improve the efficiency and effectiveness of DIRECT-type algorithms in addressing optimization challenges, particularly in scenarios with complex landscapes and varying dimensions [32,33].

The authors in [29] introduced an improved scheme by extending the set of potentially optimal hyper-rectangles for DIRECT-GL algorithm. These enhanced criteria are designed to reduce the computational cost of the algorithm by focusing on the most promising regions of the search space. By implementing the improved selection criteria, the algorithm becomes more efficient in identifying regions of interest within the optimization landscape. This leads to a reduction in the number of hyper-rectangles that need to be explored, saving computational resources and time. The enhancements introduced in this work are not limited to a specific type of problem or application. They can be applied to a wide range of optimization scenarios where DIRECT-type algorithms are utilized [30,31,34,39].

Let the partition of $\bar{D}_k^i$ at iteration k be defined as

$$\mathcal{P}_k = \{\bar{D}_k^i : i \in \mathbb{I}_k\},$$

Let $\mathbb{I}_k$ is the set of indices identifying the subsets defining the current partition $\mathcal{P}_k$. Let $\bar{\delta}_k^i$ a measure of $\bar{D}_k^i$ defined by

$$\bar{\delta}_k^i = \frac{2}{3}\|\bar{\mathbf{b}}^i - \bar{\mathbf{a}}^i\|, \tag{6}$$

Let $\mathbb{I}_k^i \subseteq \mathbb{I}_k$, represents a subset of indices that correspond to elements of $\mathcal{P}_k$ with measue δ_k^i having almost the same measure as $\bar{\delta}_k^i$ within a certain tolerance (threshold=Δ), ranging from 10^{-7} to 10^{-2}, i.e., such that $\Delta = \{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$.

$$\texttt{diff} = \left|\bar{\delta}_k^i - \delta_k^i\right| \leq \Delta, \quad i \in \mathbb{I}_k^i. \tag{7}$$

The purpose is to identify potentially optimal hyper-rectangles. It looks for hyper-rectangles (indexed by I) where the norm value (δ_k^i) is very close (within the defined tolerance) to the normalized norm value ($\bar{\delta}_k^i$).

The line 11 is used to reduce the set of potentially optimal hyper-rectangles. The code filters the hyper-rectangles and selects only those that meet a specific condition, which is having their norm value (δ_k^i) close to the normalized norm value ($\bar{\delta}_k^i$) within a tolerance of 0.0001.

In summary, this line of code helps to focus on potentially more promising hyper-rectangles, discarding those that are not as close to the desired normalized norm value. It's a way to efficiently narrow down the search space and improve the efficiency of the algorithm. An illustrative example for two different tolerance levels is given in Figure 2.

The difference between the tolerance 10^{-2} and 10^{-7} lies in the level of precision used when comparing the δ_k^i and $\bar{\delta}_k^i$ values to filter the potentially optimal hyper-rectangles.

1. Tolerance 10^{-2} :

- A tolerance of 10^{-2} (0.01) means that the algorithm will consider hyper-rectangles whose δ_k^i and $\bar{\delta}_k^i$ values are within 0.01 of each other.
- It allows for a relatively larger difference between δ_k^i and $\bar{\delta}_k^i$, meaning the algorithm will be more lenient in selecting potentially optimal hyper-rectangles.
- This might result in a larger set of potentially optimal hyper-rectangles, including some with relatively larger differences in their norm values.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

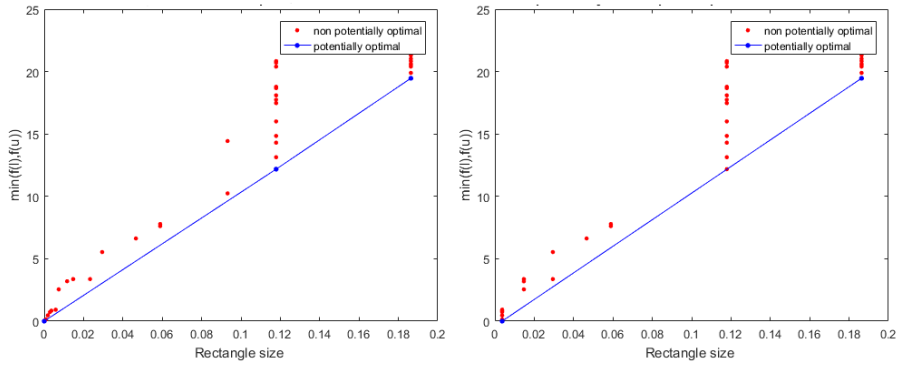

Figure 2. Grouping strategy using two different tolerance levels in the BIRECT algorithm applied to the Ackley test problem 1 at iteration 36. Small tolerance (*left side*), large tolerance (*right side*).

2. Tolerance 10^{-7} :

- A tolerance of 10^{-7} (0.0000001) means that the algorithm will consider hyper-rectangles whose δ_k^i and $\bar{\delta}_k^i$ values are within 0.0000001 of each other.
- It uses a much smaller tolerance, making the algorithm much stricter in selecting potentially optimal hyper-rectangles.
- This will result in a smaller set of potentially optimal hyper-rectangles, only including those with extremely close norm values.

The choice of tolerance depends on the specific problem and the desired level of precision in the algorithm. A larger tolerance may lead to faster execution, but it might also include some hyper-rectangles that are not truly optimal. On the other hand, a smaller tolerance will be more accurate but may require more computational effort to identify the potentially optimal hyper-rectangles. It's a trade-off between efficiency and precision in the algorithm's behavior.

Note: The algorithm assumes a zero-based index for the array elements, and the first index found satisfying the condition is returned. If no element satisfies the condition, the algorithm returns -1.

The algorithm essentially performs a linear search through the δ_k^i array and stops as soon as it finds the first element within the specified tolerance level. It's important to choose an appropriate tolerance level depending on the application and the expected values in the array.

3. Results and Discussion

3.1. Implementation

In this section we provide an overview of the methodology and objectives of our study, which involves benchmarking the new enhanced BIRECTv against the previous version of BIRECTv [3], the original BIRECT [20,21], and other DIRECT-type algorithms on a set of test problems. In our study, the size of the hyper-rectangle in BIRECTv is measured using the same measure as in the original BIRECT algorithm, while in DIRECT-l, it corresponds to the infinity norm, which allows it to collect more hyper-rectangles of the same size. This is in contrast to the Euclidean distance measure used in the original DIRECT algorithm. Our implementation uses the same set of 54 global optimization test problems from [4]. The Hedar test set is a popular benchmark for testing optimization algorithms. These problems are described in Table A1, which includes attributes like problem number, problem name, dimension, feasible domain, number of local minima, and known minimum. Some test problems have multiple variants, and the algorithm is tested for different dimensionalities. In some cases, during the initial steps of the algorithm, sampling is performed near the global minimizer. The feasible domain is modified by increasing the upper bound in these situations. these modified test problems are marked with a star. All computations were performed with MATLAB R2017b on a computer with an Intel Core i5-6300U CPU @ 3.5 GHz Processor, 8GB memory and running on Windows 10 operating

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

system. The output values are rounded up to 10 decimals. All algorithms were tested using a limit of Mmax = 500,000 function evaluations in each run. For the 54 analytical test cases with a priori known global optima f^*, the used stopping criterion is based on the percent error:

$$pe = \begin{cases} \frac{f(\bar{\mathbf{x}})-f^*}{|f^*|} \leq 10^{-4}, & f^* \neq 0, \\ f(\bar{\mathbf{x}}) \leq 10^{-4}, & f^* = 0, \end{cases} \tag{8}$$

The value of ε_{pe} was set to 10^{-4} as a default value. This value likely represents a tolerance threshold used during the optimization process such that $pe \leq \varepsilon_{\text{pe}}$.

The comparison is based on two primary criteria: the best-found function value $f(\bar{\mathbf{x}})$ and the number of function evaluations (f.eval.). These criteria help evaluate the performance of the algorithms on each test problem. The study provides statistical measures, such as averages and medians, for the number of function evaluations. The average performance provides an overall assessment of each algorithm's performance across all problems, while the median performance is less influenced by outliers and represents the middle point in the data. In the tables labeled "comparison," the best number of function evaluations is highlighted in bold font to emphasize the most efficient results. Additionally, the study includes information about the number of iterations and the execution time in seconds, these details are specifically reported on GitHub and Zenodo repositories (see Data Availability Statement below). The results of all six algorithms are reported in Table 1, where the same arguments were used: a specific domain modifications, and a grouping scheme from Algorithm 2 with a tolerence level of 10^{-4}. Note that,in our results, a correction was made during the current experiments to the minimum value acheived for the Perm test function 27 from [3]. The potentially optimal hyper-rectangles are those that have their norm values close to the normalized norm value, which means they are potentially interesting candidates for further evaluation. By filtering out the hyper-rectangles that don't satisfy this condition, the set of potentially optimal hyper-rectangles (I) is reduced to a smaller subset. These reduced hyper-rectangles are considered more interesting candidates for further evaluation or processing in the algorithm. Additionally, `BIRECTv-l`(imp.) and `BIRECTv`(imp.) are improved versions by using a special vertex database to prevent redundant sampling. Note that this assumption is not applied to the `BIRECT` algorithm, since the algorithm itself is designed to enable reuse of objective function values in descendant subregions. An illustrative example in Figure 3 demonstrates the corresponding version of the `BIRECTv` algorithm when the introduced vertex database is applied. The total number of function evaluations is 490 for `BIRECTv` and 370 for the improved version.

Algorithm 2 Find First Index within Tolerance

Require:
Input:
δ_k^i: An array of numerical values,
$\bar{\delta}_k^i$: A scalar numerical value (The normalized norm value: measure of the hyper-rectangle $\bar{D}_k^i$);
Ensure:
Output:
`index`: The index of the first occurrence in δ_k^i where the absolute difference with $\bar{\delta}_k^i$ is within the tolerance level;
procedure FIND INDEX WITHIN TOLERANCE(δ_k^i, $\bar{\delta}_k^i$)
    Set the tolerance level threshold to 0.0001 (or any desired value);
    Initialize the variable `index` to -1;
    for each `element` at index `i` in the δ_k^i array **do**
        Calculate the absolute difference `diff` between `element` and $\bar{\delta}_k^i$;
        if `diff` $\leq$ `threshold` **then**
            Set `index` to `i`;
            break
        end if
    end for
    return `index`;
end procedure

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

Table 1. Comparison between BIRECTv-l(imp.), BIRECTv(imp.), BIRECTv-l[3], BIRECTv[3], BIRECT-l(new), and BIRECT(new) algorithms.

Problem	BIRECTv-l (imp.)		BIRECTv (imp.)		BIRECTv-l [3]		BIRECTv [3]		BIRECT-l (new)		BIRECT (new)	
No.	$f(\bar{x})$	f.eval.	$f(\bar{x})$	f.eval.	$f(\bar{x})$	f.eval.	$f(\bar{x})$	f.eval.	$f(\bar{x})$	f.eval.	$f(\bar{x})$	f.eval.
1	1.22×10^{-5}	**129**	1.22×10^{-5}	153	1.22×10^{-5}	156	1.22×10^{-5}	192	1.22×10^{-5}	134	1.22×10^{-5}	158
2	1.22×10^{-5}	387	1.22×10^{-5}	1135	1.22×10^{-5}	422	1.22×10^{-5}	1578	1.22×10^{-5}	**358**	1.22×10^{-5}	1062
3	1.22×10^{-5}	1000	1.22×10^{-5}	47311	1.22×10^{-5}	1000	1.22×10^{-5}	72804	1.22×10^{-5}	**766**	1.22×10^{-5}	41654
4	8.77×10^{-5}	474	8.77×10^{-5}	742	8.77×10^{-5}	638	8.77×10^{-5}	1034	9.17×10^{-5}	**434**	9.17×10^{-5}	**434**
5	1.83×10^{-6}	**192**	1.83×10^{-6}	209	1.83×10^{-6}	254	1.83×10^{-6}	284	3.68×10^{-5}	496	3.68×10^{-5}	496
6	1.53×10^{-6}	**189**	1.53×10^{-6}	211	1.53×10^{-6}	252	1.53×10^{-6}	284	3.07×10^{-5}	682	3.07×10^{-5}	682
7	2.88×10^{-6}	**186**	2.88×10^{-6}	209	2.88×10^{-6}	248	2.88×10^{-6}	282	4.03×10^{-5}	852	4.03×10^{-5}	849
8	2.99×10^{-6}	**228**	2.99×10^{-6}	249	2.99×10^{-6}	300	2.99×10^{-6}	334	2.99×10^{-6}	330	2.99×10^{-6}	330
9	0.39791	480	0.39791	370	0.39791	652	0.39791	490	0.39790	**242**	0.39790	**242**
10	9.82×10^{-5}	1614	9.82×10^{-5}	1337	9.82×10^{-5}	2318	9.82×10^{-5}	1868	9.82×10^{-5}	**794**	9.82×10^{-5}	**794**
11	4.41×10^{-5}	263	3.18×10^{-5}	431	4.41×10^{-5}	346	3.18×10^{-5}	578	4.41×10^{-5}	**234**	4.41×10^{-5}	**234**
12	7.35×10^{-5}	**1932**	7.35×10^{-5}	2087	7.35×10^{-5}	2652	7.35×10^{-5}	2912	6.59×10^{-5}	6103	6.59×10^{-5}	6125
13	9.55×10^{-5}	28871	8.17×10^{-5}	19418	9.55×10^{-5}	38460	9.55×10^{-5}	44114	8.83×10^{-5}	**8202**	8.83×10^{-5}	**8282**
14	−0.99999	138	−0.99999	716	−0.99999	180	−0.99999	1082	−0.99999	**110**	−0.99999	558
15	3.00000	**25**	3.00000	**25**	3.00000	28	3.00000	28	3.00019	274	3.00019	274
16	4.61×10^{-7}	3440	3.697×10^{-7}	4700	4.61×10^{-7}	5192	3.697×10^{-7}	5756	7.76×10^{-7}	**3236**	9.86×10^{-3}	> 500000
17	−3.86245	**162**	−3.86245	169	−3.86245	200	−3.86245	208	−3.86243	352	3.86243	352
18	−3.32214	**490**	−3.32214	**490**	−3.32214	542	−3.32214	542	−3.32207	764	3.32207	764
19	−1.03154	**162**	−1.03154	254	−1.03154	202	−1.03154	334	−1.03154	190	1.03154	196
20	9.03×10^{-6}	103	9.03×10^{-6}	116	9.03×10^{-6}	136	9.03×10^{-6}	154	9.03×10^{-5}	**80**	9.03×10^{-6}	**80**
21	1.83×10^{-5}	388	1.83×10^{-5}	459	1.83×10^{-5}	454	1.83×10^{-5}	558	1.83×10^{-5}	**264**	1.83×10^{-5}	354
22	3.54×10^{-5}	1133	3.54×10^{-5}	6246	3.54×10^{-5}	1182	3.54×10^{-5}	7440	3.54×10^{-5}	**766**	3.54×10^{-5}	2302
23	2.71×10^{-5}	119	2.71×10^{-5}	163	2.71×10^{-5}	148	2.71×10^{-5}	208	2.71×10^{-5}	**90**	2.71×10^{-5}	**90**
24	−1.80130	142	−1.80130	231	−1.80130	184	−1.80130	314	−1.80120	136	−1.80120	**126**
25	−4.68744	5654	−4.68744	**5051**	−4.68766	8484	−4.68766	7526	−4.68757	49160	−4.68752	47196
26	−8.60559	> 500000	−7.55576	> 500000	−8.60559	> 500000	−7.55576	> 500000	−7.32708	> 500000	−7.32708	> 500000
27	0.00000	**43889**	0.0521989805	> 500000	0.00000	65536	0.00000	48724	0.00203	> 500000	0.00203	> 500000
28	4.59×10^{-5}	1837	4.59×10^{-5}	**1223**	4.59×10^{-5}	2518	4.59×10^{-5}	1624	8.34×10^{-5}	1814	4.86×10^{-5}	2108
29	9.75×10^{-5}	**2583**	9.75×10^{-5}	2867	9.75×10^{-5}	3058	9.75×10^{-5}	3400	9.12×10^{-5}	20672	9.12×10^{-5}	21260
30	0.00000	**159**	0.00000	**159**	0.00000	204	9.97×10^{-5}	40788	9.00×10^{-5}	4932	9.00×10^{-5}	5623
31	4.81×10^{-5}	523	4.81×10^{-5}	809	4.81×10^{-5}	688	4.81×10^{-5}	820	4.81×10^{-5}	**154**	4.81×10^{-5}	178
32	1.29×10^{-5}	**5237**	1.29×10^{-5}	6511	1.29×10^{-5}	8512	1.29×10^{-5}	10978	1.29×10^{-5}	66462	1.29×10^{-5}	82546
33	1.98×10^{-5}	**124**	1.98×10^{-5}	1439	1.98×10^{-5}	**124**	1.98×10^{-5}	1454	2.36×10^{-5}	1240	2.36×10^{-5}	15544
34	9.65×10^{-5}	540	9.65×10^{-5}	544	9.65×10^{-5}	700	9.65×10^{-5}	716	9.65×10^{-5}	**242**	9.65×10^{-5}	**242**
35	2.41×10^{-5}	1950	2.41×10^{-5}	2231	2.41×10^{-5}	2528	2.41×10^{-5}	3058	2.41×10^{-5}	**1494**	2.41×10^{-5}	1692
36	3.05×10^{-5}	17176	3.05×10^{-5}	27256	3.05×10^{-5}	18922	3.05×10^{-5}	31756	7.61×10^{-5}	**6104**	7.61×10^{-5}	10766
37	2.56×10^{-5}	384	1.37×10^{-5}	413	1.37×10^{-7}	486	1.37×10^{-7}	564	2.56×10^{-5}	**214**	2.56×10^{-5}	268
38	6.398×10^{-5}	17061	6.398×10^{-5}	10362	3.42×10^{-7}	25904	3.42×10^{-7}	16754	6.3981×10^{-5}	**704**	6.398×10^{-5}	3780
39	1.77×10^{-8}	**1366**	1.77×10^{-8}	55701	1.77×10^{-8}	**1366**	1.77×10^{-8}	84784	4.14×10^{-5}	2248	4.14×10^{-5}	265002
40	−10.15234	4002	−10.15234	3665	−10.15234	6146	−10.15234	5604	−10.15307	1254	−10.15307	**1220**
41	−10.40201	1536	−10.40201	1655	−10.40201	2256	−10.40201	2456	−10.402696	1186	−10.402696	**1184**
42	−10.53545	1740	−10.53545	2238	−10.53545	2476	−10.53545	3332	−10.53618	1138	−10.53618	**1108**
43	−186.72139	432	−186.72139	**181**	−186.72139	570	−186.72139	226	−186.72102	766	−186.72102	642
44	1.15×10^{-5}	**92**	1.15×10^{-5}	143	1.15×10^{-5}	112	1.15×10^{-5}	190	1.15×10^{-5}	106	1.15×10^{-5}	118
45	2.87×10^{-5}	364	2.87×10^{-5}	987	2.87×10^{-5}	392	2.87×10^{-5}	1400	2.87×10^{-5}	**294**	2.87×10^{-5}	602
46	5.74×10^{-5}	1043	5.74×10^{-5}	19418	5.74×10^{-5}	1054	5.74×10^{-5}	27566	5.74×10^{-5}	**784**	5.74×10^{-5}	8742
47	3.89×10^{-6}	348	3.89×10^{-6}	328	3.89×10^{-6}	494	3.89×10^{-6}	460	3.89×10^{-6}	226	3.89×10^{-6}	**214**
48	8.94×10^{-7}	**880**	8.94×10^{-7}	1141	8.94×10^{-7}	1102	8.94×10^{-7}	1484	3.04×10^{-5}	1006	3.04×10^{-5}	1134
49	3.28×10^{-6}	**2147**	3.28×10^{-6}	5331	3.28×10^{-6}	2452	3.28×10^{6}	6066	0.062500	> 500000	0.062500	> 500000
50	−49.99979	**1164**	−49.99979	1414	−49.99979	1312	−49.99979	1662	−49.99864	1322	−49.99864	1462
51	−209.98779	2965	−209.98779	10470	−209.98779	3114	−209.98779	11880	−209.98627	**2300**	−209.98627	3122
52	2.88×10^{-5}	122	2.88×10^{-5}	125	2.88×10^{-5}	156	2.88×10^{-5}	162	2.88×10^{-5}	**118**	2.88×10^{-5}	118
53	6.43×10^{-5}	2805	6.43×10^{-5}	2948	6.43×10^{-5}	3710	6.43×10^{-5}	3958	9.62×10^{-5}	1858	9.62×10^{-5}	**1920**
54	1.79278	> 500000	1.79278	> 500000	2.607286	> 500000	2.607286	> 500000	17.62154	> 500000	17.62154	> 500000
Average		**21488.333**		32445.204		22602.2595		27088.333		40623.833		56374.611
Median		**531.500**		1138.000		694.000		1531.000		766.000		1085.000

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

iteration:	1	fmin:	17.5082995158	f evals:	2	Iteration:	1	fmin:	17.5082995158	f evals:	2
. . .						. . .					
iteration:	20	fmin:	0.4093044620	f evals:	10	Iteration:	20	fmin:	0.4093044620	f evals:	9
. . .						. . .					
iteration:	24	fmin:	0.3994884267	f evals:	16	Iteration:	24	fmin:	0.3994884267	f evals:	13
iteration:	25	fmin:	0.3994884267	f evals:	26	Iteration:	25	fmin:	0.3994884267	f evals:	20
iteration:	26	fmin:	0.3980837400	f evals:	16	Iteration:	26	fmin:	0.3980837400	f evals:	12
iteration:	27	fmin:	0.3980837400	f evals:	14	Iteration:	27	fmin:	0.3980837400	f evals:	11
. . .						. . .					
iteration:	31	fmin:	0.3980250409	f evals:	18	Iteration:	31	fmin:	0.3980250409	f evals:	14
iteration:	32	fmin:	0.3980250409	f evals:	14	Iteration:	32	fmin:	0.3980250409	f evals:	10
iteration:	33	fmin:	0.3979818763	f evals:	24	Iteration:	33	fmin:	0.3979818763	f evals:	20
iteration:	34	fmin:	0.3979818763	f evals:	28	Iteration:	34	fmin:	0.3979818763	f evals:	18
iteration:	35	fmin:	0.3979818763	f evals:	12	Iteration:	35	fmin:	0.3979818763	f evals:	8
iteration:	36	fmin:	0.3979818763	f evals:	12	Iteration:	36	fmin:	0.3979818763	f evals:	8
iteration:	37	fmin:	0.3979818763	f evals:	22	Iteration:	37	fmin:	0.3979818763	f evals:	15
iteration:	38	fmin:	0.3979067737	f evals:	24	Iteration:	38	fmin:	0.3979067737	f evals:	15

Figure 3. An example of the iteration progress using the BIRECTv algorithm on the left-hand side from [3], and BIRECTv(imp.) on the right-hand side, while solving Branin test problem (No. 3 from Table 1).

3.2. *Discussion*

In this subsection, we discuss the performance evaluation of three optimization algorithms and their variants of the DIRECT-type that are designed for solving global optimization problems using Hedar test set [4]. All six algorithms are variations of the original BIRECT algorithm [20]. The improved versions (with "imp.") are modifications of the original BIRECTv and BIRECTv-l algorithms from [3]. Two new variations of the algorithm, "BIRECT-l (new)" and "BIRECT" (new), are also introduced. The improved version of BIRECTv-l(imp.) consistently outperform their previously published counterparts (BIRECTv-l and BIRECTv, respectively), achieving the lowest average objective function value among all six algorithms. This improvement is evident in terms of both objective function value and the number of function evaluations. This suggests that the algorithm enhancements have been successful in optimizing the problems more efficiently. However, BIRECTv-l(imp.) have the same comparable performance to BIRECTv-l in some cases, indicating that the modification of the optimization domain may not always be necessary. Similar to BIRECTv-l(imp.), the algorithm BIRECTv(imp.) generally performs well on some problems (often requires fewer function evaluations), but may not be as efficient on others as shown for problem 27. For this problem, the algorithm fails to reach a conceivable objective function contrary to BIRECTv and BIRECTv-l algorithms. The versions of BIRECTv-l and BIRECTv are evaluated based on results from [3], but with a tolerance level of 10^{-4}. For the first algorithm, both metrics (average-median) show that the algorithm is the second best algorithm. Particularly, it outperforms the BIRECTv(imp.), and dominates across all other problems. The new versions of BIRECT-l(new) and BIRECT(new) introduced in Table 2 show competitive performance compared to their predecessors [20,21], especially in terms of the number of function evaluations required. The average value is smallest using BIRECT-l(new) and BIRECT(new) from Table 2, (37230.593 and 40529.259 respectively), compared to the same algorithms from Table 1, (40623.833 and 56374.611 respectively). Even more, the average is smaller for BIRECT-l(new) (36641, 963) from Table 3 without

 doi:10.20944/preprints202311.1873.v1

the domain modification than from Table 1 with the same tolerance 10^{-4}. This means that the modification of the optimization domain is not necessary for the BIRECT algorithm. While they may not always achieve the best objective function value, they often achieve a good balance between solution quality and computational effort. The mention *f* ailed in Table 3 means that there no improvement in the objective function value during many succesive iterations, or if an increasing number of evaluations per iteration is observed. In both cases, the number of function evaluations is > 500.000. The performance of each algorithm varies across different optimization problems. Some algorithms may perform exceptionally well on certain problems but less effectively on others. This demonstrates the importance of algorithm selection based on the specific characteristics of the optimization task.

- The improved versions of BIRECTv-l and BIRECT (imp.) appear to be reliable choices for optimization tasks, as they consistently outperform the previously published versions and demonstrate competitive performance in terms of both objective value and computational effort.
- The new algorithms, BIRECT-l (new) and BIRECT (new), show promise and are particularly efficient in terms of the number of function evaluations. However, their objective function values may vary depending on the problem.
- The choice of algorithm should be problem-dependent. Some algorithms may be more suitable for specific problem characteristics, such as unimodal or multimodal objective functions, and global or local optimization.
- These informations provide a comprehensive assessment of the algorithms' performance across various aspects, including solution quality and computational efficiency.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

Table 2. Comparison between BIRECT-(new), BIRECT from [20,21], BIRECT-l-(new), and BIRECT-l from [21]

Problem	BIRECT-(new)		BIRECT	[20,21]	BIRECT-l-(new)		BIRECT-l	[21]
No.	$f(\bar{x})$	$f.eval.$	$f(\bar{x})$	$f.eval.$	$f(\bar{x})$	$f.eval.$	$f(\bar{x})$	$f.eval.$
1	2.54×10^{-5}	202	2.54×10^{-5}	202	2.54×10^{-5}	176	2.54×10^{-5}	176
2	2.54×10^{-5}	**1256**	2.54×10^{-5}	1268	2.54×10^{-5}	454	2.54×10^{-5}	454
3	2.54×10^{-5}	**45128**	2.54×10^{-5}	47792	2.54×10^{-5}	874	2.54×10^{-5}	874
4	9.17×10^{-5}	436	9.17×10^{-5}	436	9.17×10^{-5}	436	9.17×10^{-5}	436
5	4.02×10^{-5}	**468**	4.02×10^{-5}	476	4.02×10^{-5}	468	4.02×10^{-5}	468
6	3.35×10^{-5}	**472**	3.35×10^{-5}	478	3.35×10^{-5}	472	3.35×10^{-5}	472
7	3.67×10^{-5}	**474**	3.67×10^{-5}	480	3.67×10^{-5}	474	3.67×10^{-5}	474
8	6.10×10^{-5}	**188**	6.10×10^{-5}	194	6.10×10^{-5}	188	6.10×10^{-5}	188
9	0.39790	242	0.39790	242	0.39790	242	0.39790	242
10	9.82×10^{-5}	794	9.82×10^{-5}	794	9.82×10^{-5}	794	9.82×10^{-5}	794
11	4.84×10^{-5}	722	4.84×10^{-5}	722	4.84×10^{-5}	722	4.84×10^{-5}	722
12	7.15×10^{-5}	4060	7.15×10^{-5}	4060	7.15×10^{-5}	4060	7.15×10^{-5}	4060
13	9.52×10^{-5}	**161928**	9.52×10^{-5}	164826	9.52×10^{-5}	**158880**	9.52×10^{-5}	1628682
14	−0.99999	**558**	−0.99999	16420	−0.99999	**110**	−0.99999	480
15	3.00019	274	3.00019	274	3.00019	274	3.00019	274
16	7.76×10^{-7}	**4982**	7.76×10^{-7}	5106	7.76×10^{-7}	**4982**	7.76×10^{-7}	5106
17	−3.86242	352	−3.86242	352	−3.86242	352	−3.86242	352
18	−3.32206	764	−3.32206	764	−3.32206	764	−3.32206	764
19	−1.03154	**196**	−1.03154	334	−1.03154	190	−1.03154	190
20	9.09×10^{-5}	152	9.09×10^{-5}	152	9.09×10^{-5}	152	9.09×10^{-5}	152
21	1.83×10^{-5}	**968**	1.83×10^{-5}	1024	1.83×10^{-5}	**656**	1.83×10^{-5}	660
22	3.55×10^{-5}	**6402**	3.55×10^{-5}	7904	3.55×10^{-5}	1698	3.55×10^{-5}	1698
23	2.71×10^{-5}	**90**	2.71×10^{-5}	94	2.71×10^{-5}	90	2.71×10^{-5}	90
24	−1.80118	126	−1.80118	126	−1.80118	126	−1.80118	126
25	−4.68736	82562	−4.68736	**73866**	−4.68736	**101900**	−4.68736	101942
26	−7.32591	> 500000	−7.32591	> 500000	−7.32591	> 500000	−7.32591	> 500000
27	0.00203	> 500000	0.00203	> 500000	0.00203	> 500000	0.00203	> 500000
28	4.86×10^{-5}	2114	4.86×10^{-5}	2114	4.86×10^{-5}	**1820**	4.86×10^{-5}	1832
29	9.87×10^{-5}	**44950**	9.71×10^{-5}	99514	9.87×10^{-5}	**91954**	9.71×10^{-5}	92884
30	9.00×10^{-5}	**5664**	9.00×10^{-5}	10856	9.00×10^{-5}	4994	9.00×10^{-5}	**1718**
31	4.81×10^{-5}	180	4.81×10^{-5}	180	4.81×10^{-5}	156	4.81×10^{-5}	**154**
32	1.18×10^{-5}	**1162**	1.18×10^{-5}	1394	1.18×10^{-5}	474	1.18×10^{-5}	**472**
33	2.36×10^{-5}	**15658**	2.36×10^{-5}	40254	2.36×10^{-5}	1250	2.36×10^{-5}	1250
34	9.65×10^{-5}	242	9.65×10^{-5}	242	9.65×10^{-5}	242	9.65×10^{-5}	242
35	2.41×10^{-5}	**1690**	2.41×10^{-5}	1700	2.41×10^{-5}	1496	2.41×10^{-5}	**1494**
36	5.42×10^{-5}	**9100**	5.42×10^{-5}	10910	5.42×10^{-5}	4620	5.42×10^{-5}	**4590**
37	3.09×10^{-5}	236	5.64×10^{-5}	236	3.09×10^{-5}	214	5.64×10^{-5}	**210**
38	7.73×10^{-5}	**3730**	6.41×10^{-5}	7210	7.73×10^{-5}	**1074**	6.41×10^{-5}	1422
39	1.02×10^{-6}	**208670**	1.30×10^{-6}	315960	1.02×10^{-6}	**58000**	1.30×10^{-6}	58058
40	−10.15307	1272	−10.15307	**1200**	−10.15307	**1248**	−10.15307	1286
41	−10.40269	1204	−10.40269	**1180**	−10.40269	1224	−10.40269	1224
42	−10.53618	1140	−10.53618	1140	−10.53618	1162	−10.53618	**1158**
43	−186.72441	1780	−186.72441	1780	−186.72441	2114	−186.72441	2114
44	1.15×10^{-5}	118	1.15×10^{-5}	118	1.15×10^{-5}	**106**	1.15×10^{-5}	108
45	2.87×10^{-5}	**602**	2.87×10^{-5}	712	2.87×10^{-5}	294	2.87×10^{-5}	**288**
46	5.74×10^{-5}	**8742**	5.74×10^{-5}	16974	5.74×10^{-5}	784	5.74×10^{-5}	784
47	7.94×10^{-6}	**226**	7.94×10^{-6}	244	7.94×10^{-6}	226	7.94×10^{-6}	226
48	3.97×10^{-5}	**1000**	3.97×10^{-5}	1034	3.97×10^{-5}	836	3.97×10^{-5}	836
49	9.11×10^{-6}	**5538**	9.11×10^{-6}	7688	9.11×10^{-6}	3366	9.11×10^{-6}	3366
50	−49.99512	**1170**	−49.99512	1506	−49.99512	**992**	−49.99512	1138
51	−209.98007	32170	−209.98007	**30100**	−209.98007	**24704**	−209.98007	24716
52	2.88×10^{-5}	**338**	2.88×10^{-5}	502	2.88×10^{-5}	338	2.88×10^{-5}	338
53	6.44×10^{-5}	26088	6.44×10^{-5}	**20974**	6.44×10^{-5}	**27230**	6.44×10^{-5}	27364
54	9.41133	> 500000	9.41133	> 500000	9.41133	> 500000	9.41133	> 500000
Average		**40529.259**		44520.52	**37230.593**		37283.85	
Median		**1151.000**		1190.00	789.000		789.00	

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

Table 3. Number of function evaluations using BIRECT-l(new) for different values of Δ

Problem	BIRECT-l					
No./Δ	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
1	168	**166**	182	178	174	176
2	530	**448**	484	**448**	454	454
3	**840**	842	852	874	872	874
4	**370**	424	434	436	436	436
5	**328**	424	456	468	468	468
6	**328**	432	462	472	472	472
7	*failed*	*failed*	942	**474**	**474**	**474**
8	**172**	188	188	188	188	188
9	256	**242**	**242**	**242**	**242**	**242**
10	**722**	790	794	790	794	794
11	*failed*	732	**718**	722	722	722
12	*failed*	5352	**4038**	4060	4060	4060
13	*failed*	186694	**144268**	152402	156448	158880
14	**110**	**110**	**110**	**110**	**110**	**110**
15	**236**	272	274	274	274	274
16	*failed*	**2480**	3236	3452	4148	4982
17	**346**	354	352	352	352	352
18	**752**	764	764	764	764	764
19	**188**	190	190	190	190	190
20	**136**	152	152	152	152	152
21	**608**	644	656	656	656	656
22	**1590**	1698	1698	1698	1698	1698
23	**88**	90	90	90	90	90
24	**110**	126	126	126	126	126
25	**38282**	49488	90504	101900	101900	101900
26	*failed*	*failed*	*failed*	*failed*	*failed*	*failed*
27	*failed*	*failed*	*failed*	*failed*	*failed*	*failed*
28	2440	2102	**1814**	1820	1820	1820
29	*failed*	**39584**	87502	90162	92028	91954
30	*failed*	**4810**	5024	5014	5002	4994
31	**146**	152	154	156	156	156
32	**338**	436	474	474	474	474
33	**940**	1166	1240	1250	1250	1250
34	**236**	242	242	242	242	242
35	**1390**	1470	1498	1496	1496	1496
36	**4196**	4510	4612	4620	4620	4620
37	**172**	204	214	214	214	214
38	1148	1280	1400	1434	1434	**1074**
39	**41502**	49516	57452	57994	58000	58000
40	**666**	810	1254	1248	1248	1248
41	**636**	818	1186	1224	1224	1224
42	**632**	766	1138	1162	1162	1162
43	**1748**	1880	2044	2086	2114	2114
44	**104**	106	106	106	106	106
45	**286**	294	294	294	294	294
46	786	**784**	**784**	**784**	**784**	**784**
47	**202**	226	226	226	226	226
48	**728**	826	836	836	836	836
49	**2712**	3162	3332	3366	3366	3366
50	1152	**988**	992	992	992	992
51	*failed*	28268	**24578**	24704	24704	24704
52	**260**	320	338	338	338	338
53	*failed*	**25522**	27720	27286	27230	27230
54	*failed*	*failed*	*failed*	*failed*	*failed*	*failed*
Average	13121.852	44876.741	**36641, 963**	37056.407	37178.222	37230.593
Median	**725.000**	787.000	815, 000	787.000	789.000	789.000

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

4. Conclusions and Future Prospects

The paper introduced a new DIRECT-type algorithm called BIRECTv. The algorithm incorporates the most recent partitioning and selection techniques, which are essential for enhancing its performance in tackling the global optimization problems. The improvements have showed that BIRECTv outperforms existing DIRECT-type algorithms. It is more efficient and performs better in terms of convergence rates and the number of function evaluations required. Existing DIRECT-type algorithms, in contrast, tend to have slower convergence rates and often require a significantly higher number of function evaluations. They face notable difficulties when the optimal solution lies at the boundaries of feasibility. Through experimentation, the results demonstrate the algorithm's superior performance, particularly in cases where solutions are located at the boundary of feasible regions. This research has opened up new possibilities for addressing global optimization problems, which suggests that BIRECTv has the potential to make significant contributions in this field. The new algorithm is expected to have an important place among all other DIRECT-type algorithms. This implies that it could become a standard choice for solving global optimization problems with the mentioned characteristics. In conclusion, this paper sets the stage for future research by suggesting that this algorithm opens up new possibilities, which could lead to further advancements in the field.

Appendix A

Table A1. Key characteristics of the Hedar test problems [4]

Problem No.	Problem name	Dimension n	Feasible region $D = ([a_j, b_j], j = 1, \ldots, n)$	No. of local minima	Optimum f^*
$1^*, 2^*, 3^*$	Ackley	2, 5, 10	$[-15, 35]^n$	multimodal	0.0
4	Beale	2	$[-4.5, 4.5]^2$	multimodal	0.0
5^*	Bohachevsky 1	2	$[-100, 110]^2$	multimodal	0.0
6^*	Bohachevsky 2	2	$[-100, 110]^2$	multimodal	0.0
7^*	Bohachevsky 3	2	$[-100, 110]^2$	multimodal	0.0
8	Booth	2	$[-10, 10]^2$	unimodal	0.0
9	Branin	2	$[-5, 10] \times [10, 15]$	3	0.39789
10	Colville	4	$[-10, 10]^4$	multimodal	0.0
11, 12, 13	Dixon & Price	2, 5, 10	$[-10, 10]^n$	unimodal	0.0
14	Easom	2	$[-100, 100]^2$	multimodal	-1.0
15	Goldstein & Price	2	$[-2, 2]^2$	4	3.0
16^*	Griewank	2	$[-600, 700]^2$	multimodal	0.0
17	Hartman	3	$[0, 1]^3$	4	-3.86278
18	Hartman	6	$[0, 1]^6$	4	-3.32237
19	Hump	2	$[-5, 5]^2$	6	-1.03163
20, 21, 22	Levy	2, 5, 10	$[-10, 10]^n$	multimodal	0.0
23^*	Matyas	2	$[-10, 15]^2$	unimodal	0.0
24	Michalewics	2	$[0, \pi]^2$	2!	-1.80130
25	Michalewics	5	$[0, \pi]^5$	5!	-4.68765
26	Michalewics	10	$[0, \pi]^{10}$	10!	-9.66015
27	Perm	4	$[-4, 4]^4$	multimodal	0.0
28, 29	Powell	4, 8	$[-4, 5]^n$	multimodal	0.0
30	Power Sum	4	$[0, 4]^4$	multimodal	0.0
$31^*, 32^*, 33^*$	Rastrigin	2, 5, 10	$[-5.12, 6.12]^n$	multimodal	0.0
34, 35, 36	Rosenbrock	2, 5, 10	$[-5, 10]^n$	unimodal	0.0
$37, 38, 39^*$	Schwefel	2, 5, 10	$[-500, 500]^n$	unimodal	0.0
40	Shekel, $m = 5$	4	$[0, 10]^4$	5	-10.15320
41	Shekel, $m = 7$	4	$[0, 10]^4$	7	-10.40294
42	Shekel, $m = 10$	4	$[0, 10]^4$	10	-10.53641
43	Shubert	2	$[-10, 10]^2$	760	-186.73091
$44^*, 45^*, 46^*$	Sphere	2, 5, 10	$[-5.12, 6.12]^n$	multimodal	0.0
$47^*, 48^*, 49^*$	Sum squares	2, 5, 10	$[-10, 15]^n$	unimodal	0.0
50	Trid	6	$[-36, 36]^6$	multimodal	-50.0
51	Trid	10	$[-100, 100]^{10}$	multimodal	-210.0
$52^*, 53^*, 54^*$	Zakharov	2, 5, 10	$[-5, 11]^n$	multimodal	0.0

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2023 doi:10.20944/preprints202311.1873.v1

Table A2. Comparison between BIRECT-(new), BIRECT, DIRECT-l, and DIRECT

Problem	BIRECT-(new)		BIRECT		DIRECT-l		DIRECT	
No.	$f(\bar{x})$	$f.eval.$	$f(\bar{x})$	$f.eval.$	$f(\bar{x})$	$f.eval.$	$f(\bar{x})$	$f.eval.$
1	2.54×10^{-5}	202	2.54×10^{-5}	202	7.53×10^{-5}	**135**	7.53×10^{-5}	255
2	2.54×10^{-5}	**1256**	2.54×10^{-5}	1268	7.53×10^{-5}	1777	7.53×10^{-5}	8845
3	2.54×10^{-5}	**45128**	2.54×10^{-5}	47792	3.57445	> 500000	7.53×10^{-5}	80927
4	9.17×10^{-5}	436	9.17×10^{-5}	436	9.29×10^{-5}	**247**	9.29×10^{-5}	655
5	4.02×10^{-5}	468	4.02×10^{-5}	476	3.09×10^{-6}	**205**	3.09×10^{-5}	327
6	3.35×10^{-5}	472	3.35×10^{-5}	478	2.58×10^{-6}	**233**	2.58×10^{-5}	345
7	3.67×10^{-5}	**474**	3.67×10^{-5}	480	8.21×10^{-5}	573	8.21×10^{-5}	693
8	6.10×10^{-5}	**188**	6.10×10^{-5}	194	6.58×10^{-5}	215	6.58×10^{-5}	295
9	0.39790	242	0.39790	242	0.39789	**159**	0.39789	195
10	9.82×10^{-5}	**794**	9.82×10^{-5}	**794**	3.83×10^{-5}	3379	6.08×10^{-5}	6585
11	4.84×10^{-5}	722	4.84×10^{-5}	722	5.32×10^{-5}	**485**	6.25×10^{-5}	513
12	7.15×10^{-5}	**4060**	7.15×10^{-5}	**4060**	6.45×10^{-5}	54843	6.45×10^{-5}	19661
13	9.52×10^{-5}	**161928**	9.52×10^{-5}	164826	0.66667	> 500000	5.79×10^{-5}	372619
14	−0.99999	**558**	−0.99999	16420	−0.99999	6851	−0.99999	32845
15	3.00019	274	3.00019	274	3.00009	**115**	3.00009	191
16	7.76×10^{-7}	**4982**	7.76×10^{-7}	5106	4.84×10^{-6}	8379	4.84×10^{-6}	9215
17	−3.86242	352	−3.86242	352	−3.86245	**111**	−3.86245	199
18	−3.32206	764	−3.32206	764	−3.32207	**295**	−3.32207	571
19	−1.03154	196	−1.03154	334	−1.03162	**137**	−1.03162	321
20	9.09×10^{-5}	152	9.09×10^{-5}	152	2.10×10^{-5}	**77**	2.10×10^{-5}	105
21	1.83×10^{-5}	968	1.83×10^{-5}	1024	3.65×10^{-5}	**359**	3.65×10^{-5}	705
22	3.55×10^{-5}	6402	3.55×10^{-5}	7904	3.55×10^{-5}	**5297**	6.23×10^{-5}	5589
23	2.71×10^{-5}	90	2.71×10^{-5}	94	3.81×10^{-5}	**71**	3.81×10^{-5}	107
24	−1.80118	126	−1.80118	126	−1.80127	**45**	−1.80127	69
25	−4.68736	82562	−4.68736	73866	−4.68721	26341	−4.68721	**13537**
26	−7.32591	> 500000	−7.32591	> 500000	−7.84588	> 500000	−7.87910	> 500000
27	0.00203	> 500000	0.00203	> 500000	0.04054	> 500000	0.04355	> 500000
28	4.86×10^{-5}	**2114**	4.86×10^{-5}	**2114**	6.52×10^{-5}	32331	9.02×10^{-5}	14209
29	9.87×10^{-5}	**44950**	9.71×10^{-5}	99514	0.02488	> 500000	0.02142	> 500000
30	9.00×10^{-5}	**5664**	9.00×10^{-5}	10856	0.03524	> 500000	0.00215	> 500000
31	4.81×10^{-5}	**180**	4.81×10^{-5}	**180**	2.30×10^{-5}	1727	2.30×10^{-5}	987
32	1.18×10^{-5}	**1162**	1.18×10^{-5}	1394	4.97479	> 500000	4.97479	> 500000
33	2.36×10^{-5}	**15658**	2.36×10^{-5}	40254	5.01600	> 500000	9.94967	> 500000
34	9.65×10^{-5}	**242**	9.65×10^{-5}	**242**	9.65×10^{-5}	285	9.65×10^{-5}	1621
35	2.41×10^{-5}	**1690**	2.41×10^{-5}	1700	5.75×10^{-5}	2703	8.80×10^{-5}	20025
36	5.42×10^{-5}	**9100**	5.42×10^{-5}	10910	8.29×10^{-5}	74071	8.29×10^{-5}	174529
37	3.09×10^{-5}	**236**	5.64×10^{-5}	**236**	2.88×10^{-5}	341	2.88×10^{-5}	255
38	7.73×10^{-5}	**3730**	6.41×10^{-5}	7210	7.21×10^{-5}	322039	7.21×10^{-5}	31999
39	1.02×10^{-6}	**208670**	1.30×10^{-6}	315960	1269.34444	> 500000	1187.63199	> 500000
40	−10.15307	1272	−10.15307	1200	−10.15234	**147**	−10.15234	155
41	−10.40269	1204	−10.40269	1180	−10.40196	**141**	−10.40196	145
42	−10.53618	1140	−10.53618	1140	−10.53539	**139**	−10.53539	145
43	−186.72441	**1780**	−186.72441	**1780**	−186.72153	2043	−186.72153	2967
44	1.15×10^{-5}	118	1.15×10^{-5}	118	8.74×10^{-5}	**91**	8.74×10^{-5}	209
45	2.87×10^{-5}	602	2.87×10^{-5}	712	7.49×10^{-5}	**465**	9.39×10^{-5}	4653
46	5.74×10^{-5}	8742	5.74×10^{-5}	16974	9.63×10^{-5}	**2057**	6.32×10^{-5}	99123
47	7.94×10^{-6}	226	7.94×10^{-6}	244	3.53×10^{-5}	**77**	3.52×10^{-5}	107
48	3.97×10^{-5}	1000	3.97×10^{-5}	1034	7.19×10^{-5}	**411**	7.19×10^{-5}	833
49	9.11×10^{-6}	5538	9.11×10^{-6}	7688	7.76×10^{-6}	**1809**	7.76×10^{-5}	8133
50	−49.99512	**1170**	−49.99512	1506	−49.99525	8731	−49.99525	5693
51	−209.98007	32170	−209.98007	**30100**	−209.92644	> 500000	−209.98085	90375
52	2.88×10^{-5}	338	2.88×10^{-5}	502	7.95×10^{-5}	**209**	7.95×10^{-5}	237
53	6.44×10^{-5}	26088	6.44×10^{-5}	**20974**	0.11921	> 500000	9.71×10^{-5}	316827
54	9.41133	> 500000	9.41133	> 500000	16.47703	> 500000	28.96394	> 500000
Average		**40529.26**		44520.52	121484.19		98677.70	
Median		**1151.00**		1190.00	1752.00		3810.00	

Author Contributions: Conceptualization, L.C.; Data curation, L.C.; Formal analysis, L.C. and M.L.; Funding acquisition, M.L.; Investigation, N.B. and L.C.; Methodology, N.B. and L.C.; Project administration, M.L. and L.C.; Software, N.B. and L.C.; Supervision, L.C. and M.L.; Validation, L.C. and M.L.; Writing—original draft, L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data underlying this article are available on GitHub repository from `BIRECTv` v1.1.0 - https://github.com/lchiter/Algorithm-BIRECTv/releases (accessed on 20 September 2023), and used under the MIT license, or at Zenodo: https://zenodo.org/record/7416231 (accessed on 20 July 2023). The first codes for the algorithms `BIRECT`(new) and `BIRECT-l`(new) are made available from https://data.mendeley.com/datasets/t6vv9yknbc/1.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Floudas, C.A.: Deterministic Global Optimization: Theory, Methods and Applications. *Nonconvex Optimization and Its Applications*, vol. 37. Springer, Boston, MA (1999). https://doi.org/10.1007/978-1-4757-4949-6
2. Gablonsky, J.M., Kelley, C.T.: A locally-biased form of the DIRECT algorithm. *J. of Glob. Optim.* (2001), 21(1), 27-37. DOI:10.1023/A:1017930332101
3. Guessoum, N., Chiter, L.: Diagonal Partitioning Strategy Using Bisection of Rectangles and a Novel Sampling Scheme. *MENDEL*.(2023), 29, 2 131-146. :https://doi.org/10.13164/mendel.2023.2.131.
4. Hedar, A.: Test functions for unconstrained global optimization. http://www-optima.amp.i.kyotou.ac.jp/member/student/hedar/Hedar_files/TestGO.htm (2005). (accessed on 23 August 2006)
5. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization. *Nonconvex Optimization and Its Application*. Kluwer Academic Publishers (1995)
6. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches. Springer, Berlin (1996)
7. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. *J. of Optim. Theory and Appl.* (1993), 79(1), 157-181. DOI:10.1007/BF00941892
8. Jones, D.R.: The Direct global optimization algorithm. In: C.A. Floudas, P.M. Pardalos (eds.) *The Encyclopedia of Optimization*, pp. (2001), 431-440. Kluwer Academic Publishers, Dordrect (2001)
9. Jones, D.R., Martins, J.R.R.A.: The DIRECT algorithm: 25 years later. *J. Glob. Optim.* 79, 521–566 (2021). https://doi.org/10.1007/s10898-020-00952-6
10. Ma, K., Rios, L. M., Bhosekar, A., Sahinidis, N., V., Rajagopalan, S.: Branch-and-Model: a derivative-free global optimization algorithm. *Computational Optimization and Applications*. (2023), https://doi.org/10.1007/s10589-023-00466-3
11. Kvasov, D.E., Sergeyev, Y.D.: Lipschitz gradients for global optimization in a one-point-based partitioning scheme. Journal of Computational and Applied Mathematics. (2012), 236(16), 4042-4054. DOI:10.1016/j.cam.2012.02.020
12. Liberti, L., Kucherenko, S.: Comparison of deterministic and stochastic approaches to global optimization. International Transactions in Operational Research 12(3), 263–285 (2005) https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1475-3995.2005.00503.x.https://doi.org/10.1111/j.1475-3995.2005.00503.x
13. Liu, H., Xu, S.,Wang, X.,Wu, J., Song, Y.: A global optimization algorithm for simulation-based problems via the extended DIRECT scheme. Eng. Optim. (2015), 47(11), 1441–1458. DOI:10.1080/0305215X.2014.971777
14. Liu, Q., Zeng, J., Yang, G.: MrDIRECT: a multilevel robust DIRECT algorithm for global optimization problems. Journal of Global Optimization. (2015), 62(2), 205-227. DOI:10.1007/s10898-014-0241-8
15. Liuzzi, G., Lucidi, S., Piccialli, V.: Exploiting derivative-free local searches in direct-type algorithms for global optimization. Computational Optimization and Applications pp. (2014), 1-27. DOI:10.1007/s10589-015-9741-9
16. Paulavičius, R., Žilinskas, J., Grothey, A.: Parallel branch and bound for global optimization with combination of Lipschitz bounds. *Optimization Methods and Software*. (2011), 26(3), 487-498. DOI:10.1080/10556788.2010.551537
17. Paulavičius, R., Žilinskas, J.: Simplicial Global Optimization. *SpringerBriefs in Optimization*. Springer New York, New York, NY (2014). DOI:10.1007/978-1-4614-9093-7
18. Paulavičius, R., Sergeyev, Y.D., Kvasov, D.E., Zilinskas, J.: Globally-biased DISIMPL algorithm for expensive global optimization. J. Glob. Optim. (2014) 59, 545–567. DOI:10.1007/s10898-014-0180-4
19. Paulavičius, R.; Zilinskas, J. Simplicial Lipschitz optimization without the Lipschitz constant. *J. Glob. Optim.* **2014**, 59, 23–40. doi:10.1007/s10898-013-0089-3
20. Paulavičius, R., Chiter, L., Žilinskas, J.: Global optimization based on bisection of rectangles, function values

at diagonals, and a set of Lipschitz constants. J. Glob. Optim. (2018), 71(1), 5–20. DOI:10.1007/s10898-016-0485-6
21. Paulavičius, R., Sergeyev, Y.D., Globally-biased BIRECT algorithm with local accelerators for expensive global optimization, Expert Systems with Applications. November 2019.
22. Sergeyev, Y.D.: An efficient strategy for adaptive partition of N-dimensional intervals in the framework of diagonal algorithms. Journal of Optimization Theory and Applications. (2000), 107(1), 145-168. DOI:10.1023/A:1004613001755
23. Sergeyev, Y.D.: Efficient partition of n-dimensional intervals in the framework of one-point-based algorithms. Journal of optimization theory and applications. (2005), 124(2), 503-510. DOI:10.1007/s10957-004-0948-7
24. Sergeyev, Y.D., Kvasov, D.E.: Global search based on diagonal partitions and a set of Lipschitz constants. SIAM Journal on Optimization. (2006), 16(3), 910-937. DOI:10.1137/040621132
25. Sergeyev, Y.D., Kvasov, D.E.: Diagonal Global Optimization Methods. FizMatLit, Moscow (2008). In Russian
26. Sergeyev, Y.D., Kvasov, D.E.: On deterministic diagonal methods for solving global optimization problems with Lipschitz gradients. In: Optimization, Control, and Applications in the Information Age, 130, pp . Springer International Publishing Switzerland. (2015), 315-334. DOI:10.1007/978-3-319-18567-5-16
27. Sergeyev, Y.D., Kvasov, D.E.: Lipschitz global optimization. In: Cochran, J.J., Cox, L.A., Keskinocak, P., Kharoufeh, J.P., Smith, J.C. (eds.) Wiley Encyclopedia of Operations Research and Management Science (in 8 Volumes) vol. 4, pp. 2812–2828. John Wiley and Sons, New York, NY, USA (2011)
28. Sergeyev, Y.D.; Kvasov, D.E. Deterministic Global Optimization: An Introduction to the Diagonal Approach; SpringerBriefs in Optimization; Springer: Berlin, Germany, 2017. https://doi.org/10.1007/978-1-4939-7199-2.
29. Stripinis, L., Paulavičius, R., Žilinskas, J.: Improved scheme for selection of potentially optimal hyperrectangles in DIRECT. Optim. Lett. (2018), 12(7), 1699–1712. DOI:10.1007/s11590-017-1228-4
30. Stripinis, L., Paulavičius, R.: DIRECTGOLib - DIRECT Global Optimization test problems Library, v1.1. Zenodo (2022). https://doi.org/10.5281/zenodo.6491951
31. Stripinis, L., Kůdela, J., Paulavičius, R.: Directgolib - direct global optimization test problems library (2023). https://github.com/blockchain-group/DIRECTGOLib.Pre-releasev2.0
32. Stripinis, L., Paulavičius, R. Novel Algorithm for Linearly Constrained Derivative Free Global Optimization of Lipschitz Functions; *Mathematics* , 11(13), (2023), 2920. https://doi.org/10.3390/math11132920.
33. Stripinis, L., Paulavičius, R. GENDIRECT: a GENeralized DIRECT-type algorithmic framework for derivative-free global optimization. https://doi.org/10.48550/arXiv.2309.00835.
34. Stripinis, L., Paulavičius, R.: DIRECTGO: A new DIRECT-type MATLAB toolbox for derivative free global optimization. GitHub (2022). https://github.com/blockchain-group/DIRECTGO
35. Stripinis, L., Paulavičiuss, R.: DIRECTGO: A new DIRECT-type MATLAB toolbox for derivative free global optimization. arXiv (2022). https://arxiv.org/abs/2107.0220
36. Stripinis, L., Paulavičius, R.: Lipschitz-inspired HALRECT Algorithm for Derivative-free Global Optimization. https://doi.org/10.48550/arXiv.2205.03015
37. Stripinis, L.; Paulavičius, R. An extensive numerical benchmark study of deterministic vs. stochastic derivative-free global optimization algorithms. https://doi.org/10.48550/ARXIV.2209.05759.
38. Stripinis, L.; Paulavičius, R. An empirical study of various candidate selection and partitioning techniques in the DIRECT framework. J. Glob. Optim. 2022, 1–31. https://doi.org/10.1007/s10898-022-01185-5
39. Stripinis, L. Improvement, development and implementation of derivative-free global optimization algorithms. DOCTORAL DISSERTATION, VILNIUS UNIVERSITY, 2001, https://doi.org/10.15388/vu.thesis.138
40. Tsvetkov, E.A., Krymov, R.A. Pure Random Search with Virtual Extension of Feasible Region. J Optim Theory Appl 195, 575—595 (2022). https://doi.org/10.1007/s10957-022-02097-w
41. Tuy, H. Convex Analysis and Global Optimization. Springer Science & Business Media (2013)
42. Zhigljavsky, A., Žilinskas, A. Stochastic Global Optimization. Springer, New York (2008)